Why Johnny the Developer Can't Work with Public Key Certificates*

An Experimental Study of OpenSSL Usability

Martin Ukrop[0000−0001−8110−8926] and Vashek Matyas

Centre for Research on Cryptography and Security
Faculty of Informatics, Masaryk University, Czechia
mukrop@mail.muni.cz, matyas@fi.muni.cz

Abstract. There have been many studies exposing poor usability of security software for the common end user. However, only a few inspect the usability challenges faced by more knowledgeable users. We conducted an experiment to empirically assess usability of the command line interface of OpenSSL, a well known and widely used cryptographic library. Based on the results, we try to propose specific improvements that would encourage more secure behavior. We observed 87 developers/administrators at two certificate-related tasks in a controlled environment. Furthermore, we collected participant opinions on both the tool interface and available documentation. Based on the overall results, we deem the OpenSSL usability insufficient according to both user opinions and standardized measures. Moreover, the perceived usability seems to be correlated with previous experience and used resources. There was a great disproportion between the participant views of a successful task accomplishment and the reality. A general dissatisfaction with both OpenSSL interface and its manual page was shared among the majority of the participants. As hinted by a participant, OpenSSL gradually "turned into a complicated set of sharp kitchen knives" – it can perform various jobs very well, but laymen risk stabbing themselves in the process. This highlights the necessity of a usable design even for tools targeted at experienced users.[1]

1 Introduction

The first users of any newly created software are its own developers and testers. For such knowledgeable users, one would therefore not expect usability failures similar to those exposed to the "common Johnny" [14,23,27]. We have conducted an experiment to empirically assess usability of the command line interface of OpenSSL, a widely-used cryptographic library. Outcomes can be briefly summarized by quoting one of the study participants: *"I am surprised that even as a crypto expert I am unable to use OpenSSL."* Not only is the software barely

* This is the author's version. The final publication will be available at Springer via http://dx.doi.org/10.1007/978-3-319-76953-0_3

[1] Supplementary material available at crocs.fi.muni.cz/papers/rsa2018.

usable for other developers – results indicate that the perceived usability even *decreases* with IT experience gain.

OpenSSL is an open source project providing a full-featured commercial-grade toolkit for SSL/TLS and general-purpose cryptography [5]. As of 2017, it is by far the best known and the most used library for generating and manipulating public key certificates [21]. Even though it may be superseded by specialized tools in some cases (e.g., Certbot client by Let's Encrypt[2] for obtaining, deploying and refreshing server certificates), it is still a leading universal tool. Last but not least, there is plenty anecdotal evidence of its poor usability, again quoting one of the study participants: *"Working with OpenSSL is a struggle every time – it takes at least 20-30 minutes to find something."* A digest of other representative quotations can be found in Appendix B.

Our pilot study (see Section 2 for details) compared the interfaces of three similar cryptographic tools – suggesting that OpenSSL, although having the best rating of the three, is still hard to use even for knowledgeable developers.

To rigorously inspect OpenSSL usability, we designed and performed an experiment through a research booth at a developer conference. Attendees could help the research by accomplishing two tasks – using command line OpenSSL to generate a self-signed X.509 certificate and to validate a set of certificates with the same tool. We analyzed participant success, OpenSSL usability, resources used during task completion, security-related behavior and their opinions.

The overall usability score for command line OpenSSL indicated a rather poor user experience. Only 16% of the participants considered the OpenSSL interface OK. Complaints included being too complex, too low-level, not following Linux conventions and having bad and/or confusing structure. A similar proportion was satisfied with the manual page, complaints ranging from missing examples, through confusing structure, to style being only for experts (particulars in Section 4.5). As mentioned earlier, OpenSSL usability seems to decrease with users gaining more experience (moving from school to work, working in the field longer, getting to know other tools), see Section 4.3. Furthermore, only 45% of the participants successfully created a valid self-signed certificate – this is in a sharp contrast with the subjective assessment of the participants, in which over 87% claimed to have succeeded (Section 4.1). Regarding the resources used during task completion, about half the participants used a combination of informal online sources (tutorials, blogs, forums) and official man pages installed locally (Section 4.5).

This work has *three main contributions* to the usable security research and wider developer community:

1. It constitutes one of the first reasonably-sized studies of OpenSSL usability.
2. It presents an empirical analysis of developers' behavior combined with their opinions when accomplishing security-related tasks.
3. It proposes specific and feasible suggestions for OpenSSL improvements.

The paper is organized as follows: After the introduction, Section 2 briefly outlines our earlier pilot experiment. Section 3 then describes the main experiment,

[2] Let's Encrypt is a free, automated and open certificate authority, see letsencrypt.org.

namely details of the tasks and participant background. Results and observations are presented in Section 4, with study limitations in Section 5. Section 6 gives accounts of the related research and Section 7 concludes the paper.

2 The Pilot Experiment

Before the main experiment, we conducted a pilot study with 26 Master-level students focused on IT security. The aim was to compare the usability of three similar command line tools for manipulating X.509 certificates (GnuTLS, NSS and OpenSSL). It was a within-subjects experiment (each participant using all three tools in succession) with the same tasks as described in the next section (certificate generation/validation).

Both the numerical usability ratings and students' self-reported sentiment towards the libraries imply OpenSSL is superior to GnuTLS that, in turn, fared better than NSS. Despite the low usability score and user complaints, OpenSSL seems to be not only the most wide-spread tool but also the one with a relatively reasonable usability (when compared to alternatives). For more details, see [16].

Although the pilot study was very similar to the main experiment, the conclusions may not be directly applicable. Firstly, the participant population was different – security-oriented students vs. a heterogeneous group of developers. Furthermore, the students used OpenSSL in courses (although for different tasks). Secondly, the task success and usability ratings may have been skewed by the participants using multiple tools for the same tasks – we tried to eliminate this by counterbalancing (randomizing) the tool order.

3 The Main Experiment Settings

The experiment took place at DevConf,[3] where the conference participants were asked to complete one or two simple X.509 certificate-related tasks advertised to take about 30 minutes. We did not give any financial compensation for participation, only a branded winter cap.

Each participant was provided with a computer running virtualized Ubuntu 16.04 with OpenSSL 1.0.2g, recording the screen, browsing history and terminal input/output. Before attempting the tasks, each participant filled in a questionnaire on their previous experience. After the experiment, there was a short semi-structured interview, concluded by answering standardized questions on OpenSSL usability. The course of our experiment is summarized in Fig. 1. The questionnaire and interview outline are in Appendix A.

All participants were briefed about the extent of processed personal information and signed an informed consent form before starting the experiment. The data was collected anonymously. The study design has been approved by the Research Ethics Committee of Masaryk University.

[3] DevConf is an annual conference for developers, admins and users of open source technologies organized by Red Hat Czech with about 1500 attendees, see devconf.cz.

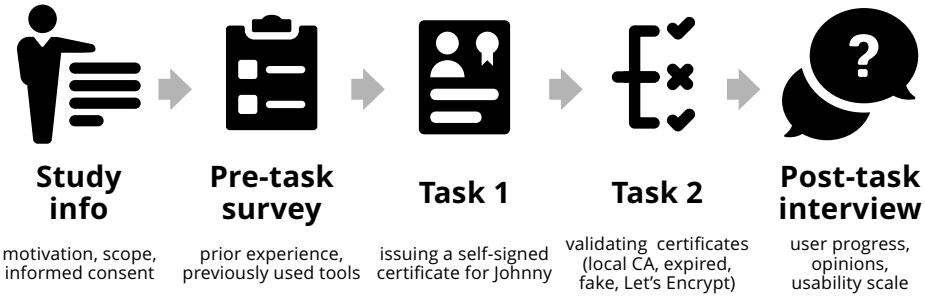

Fig. 1. The experiment core consisted of 2 certificate-related tasks, preceded and followed by short participant surveys.

3.1 Tasks

In both tasks, each participant was explicitly asked to use `openssl`, the command line utility provided by the OpenSSL project [5]. It was emphasized that they can use all common resources: read the documentation, search for examples online, browse online forums, etc.

Task 1: Issuing Certificates. The first task puts the participant into the position of a software tester. The tested application was said to have an option to load a public key certificate. The participant was further instructed that to test it, they should generate a new public key certificate for the user Johnny.

The task aimed at generating a self-signed certificate (although a pair of a certificate authority (CA) certificate and an end-point certificate would also be a viable option). The certificate may have been generated with or without the intermediate certificate signing request (CSR). Furthermore, the keypair might have been generated separately or during the certificate/CSR generation. These two choices are independent of each other, resulting in four different ways of certificate creation. Moreover, the process of setting the subject attributes could have been interactive or not (providing the information as command line arguments).

Task 2: Validating Certificates. The second task presented a similar setup: The participant (in a work environment again) was asked by their team lead to validate four certificates they got from their partners. They were reminded not to forget that, except for the system-installed CAs, they trust also the company internal testing authority provided in a local file.

The task required the user to verify 4 certificates, correctly specifying a local trusted CA and taking into account the default installed CAs. The provided certificates were as follows:

1. A *valid* certificate issued by the *local CA*.
2. An *expired* certificate issued by the *local CA*.

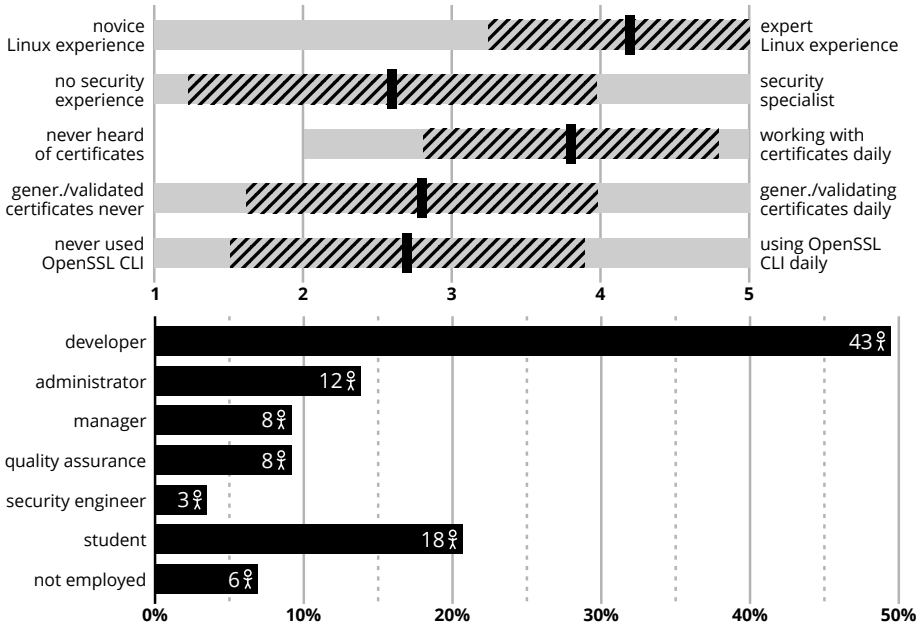

Fig. 2. The profile of all 87 experiment participants (scale questions display range in gray, mean and standard deviation; job positions are not exclusive).

3. A *fake* certificate pretending to be from the *local CA* (bad signature).
4. A *valid* certificate issued by *Let's Encrypt CA*.

Even though OpenSSL verifies against system CAs by default, this fact is not trivial to find out. If unsure, the participant could have explicitly provided the path to the default CA database.

3.2 Participants

87 participants (from now on the symbol ♀ is used) took part in the experiment. Since all were attendees of a developer conference, we expected a considerable (although very variable) background knowledge of IT or even certificate generation/validation. To investigate the relationship of the prior experience to results, we mapped these using a pre-task questionnaire (see Appendix A).

All participants in the study were male (not intentionally). On average, they had been in IT for a bit over 12 years (study+work). Nearly half the participants described themselves as developers, only 3 participants explicitly stated being focused on security. In general, the (self-reported) Linux experience was very high as well as the (self-reported) awareness of what public key certificates are and what they are used for. For averages and standard deviations, see Fig. 2.

The last part of the questionnaire inquired about tools the participants had used prior to the experiment. OpenSSL, being the most common, was used by

82% of the participants (71 [4]). The second most popular tool was NSS [4] (16%, 14), followed by GnuTLS [6] (10%, 9) and Java Keytool [2] (9%, 8). Nearly a quarter of the respondents (24%, 21) mentioned still other tools. 15% of the participants (13) had never used any of these tools before.

Exploring the relationships among the variables in previous experience (number of years in IT, Linux experience, security background, domain knowledge, certificate experience, OpenSSL usage), we see that all pairs are significantly[5] correlated (Spearman's rank-order coefficient[6] $\rho \approx 0.5$). The largest correlation is between the previous experience with generating/validating certificates and using OpenSSL ($\rho = 0.776$). This confirms the general opinion of OpenSSL being a common tool for manipulating certificates.

4 Results and Observations

In this section, we report the summary of participant success (Sections 4.1 and 4.2), the perceived tool usability (Section 4.3), noteworthy user behaviors (Section 4.4) and the resources used to accomplish the tasks (Section 4.5).

4.1 Task Success

Task 1: Issuing Certificates. We differentiate five levels of success based on what the user generated:

Johnny certificate	39 45%	A valid self-signed certificate containing Johnny (or a similar string) in at least one of the subject fields.
Certificate	23 26%	A valid self-signed certificate not mentioning Johnny (technically OK, but the task specifically asked for a certificate for "user Johnny").
CSR	3 3%	A valid certificate signing request.
Keypair	17 20%	An asymmetric keypair generated by OpenSSL.
Nothing	5 6%	Nothing or unrelated files (e.g., an SSH keypair).

Only 45% of the participants (39) successfully created a valid certificate mentioning Johnny in the subject. This is in sharp contrast with the subjective assessment of the participants, in which over 87% (76) claimed to have succeeded in the task (7%, 6 knew they failed and 6%, 5 were unsure).

Taking the success as a discrete ordinal scale, the results have a small statistically significant correlation with the Linux experience ($\rho = 0.26$), prior experience with generating certificates ($\rho = 0.23$), prior OpenSSL experience ($\rho = 0.28$)

[4] If not stated otherwise, the presented analyses include all 87 participants.
[5] All presented results are statistically significant with a confidence level of $\alpha = 5\%$.
[6] Spearman's rank-order coefficient ρ can assume values from -1 to 1, the sign indicating the direction of the relationship and the absolute value indicating the intensity from 0 (no relationship) to 1 (perfect linear relationship). [24]

and with the number of years the participant has studied/worked in IT ($\rho = 0.29$). This is in accordance with what we expected – the more practical experience the user has, the higher is the probability of him generating the certificate correctly. Note the task success was significantly correlated with neither the general knowledge of certificate principles nor with security experience.

Task 2: Validating Certificates. Since not all participants had enough time for both tasks (due to conference schedule), only 72/87 attempted the second task. This time the success categorization is based on the way respondents performed the validation:

Explicit	14/72 19%	Correct OpenSSL command explicitly checking both the local and system-installed CAs.
Implicit	51/72 71%	Correct command setting only the local CA.
Incomplete	4/72 6%	Verification command with incorrect trust settings (e.g., setting the local CA as not trusted).
Visual	3/72 4%	Not verifying the signature, only visually comparing the issuer and subject in the certificates.

The second case (*Implicit*) is also considered a complete success since OpenSSL automatically checks against some default trust store (OS-dependent). However, this fact is rather complicated to find both online and in the official documentation. All the study participants who checked whether this is the default behavior ended up doing the validation explicitly. That is why we consider the categories separately. Only 19% (14/72) did the explicit validation.

Inspecting relationships of the task success with the prior experience, we see small statistically significant correlations with the Linux experience ($\rho = 0.36$), prior experience with generating certificates ($\rho = 0.22$), prior OpenSSL experience ($\rho = 0.30$) and with the number of years in IT ($\rho = 0.30$). In contrast with Task 1, this time there are correlations with both the theoretical knowledge ($\rho = 0.32$) and security experience ($\rho = 0.23$, $p = 0.057$). We hypothesize this is because validating certificates requires more detailed knowledge (PKI trust model), lacks the interactivity of the generation process and presents a much more cryptic error messages (see Section 4.3).

4.2 Created Certificates

This section presents statistics of the created certificates. For keysize, we consider everyone succeeding in creating at least a keypair (82), for subject fields at least a CSR (65) and for other features only respondents creating a certificate (62).

Keysize. Nearly all participants (98%, 80/82) created a standard RSA key – the remaining 2 users generated an elliptic curve key. Even though none of the possible solutions require the user to explicitly state the bitsize of the key (there

is always an applicable default), most of the users did so (85%, 70/82 🎗). This is mostly due to the fact that nearly all available tutorials and examples specify the keysize explicitly (see Section 4.5). In nearly half the cases (42%, 34/80 🎗) a 2048-bit key was generated, followed by 4096-bit key (38%, 30/80 🎗). The remaining 20% (16/80 🎗) created a 1024-bit key. It is positive that the weakest keysize (considered inadequate in 2017 [9]) was created by the smallest group of participants, but it is still a non-trivial fraction.

Subject Fields. As nearly all the subject fields offer the possibility of non-validated text input, this was the aspect that differed the most among the created certificates. Most notably, 42% of the certificates (27/65 🎗) have the organization field set to *Internet Widgits Pty Ltd* – this nonsensical value is the default in the interactive process. Its usage is greatly enhanced by the way the to-be-set default is displayed (a few participants that inspected their own certificate after creation were quite surprised to find it instead of an empty field). From the 95% users (62/65 🎗) using the interactive subject-setting process, 35% (22/62 🎗) filled in all the fields they were offered. This hints that the selection of fields displayed during the interactive setup greatly influences what fields will be populated by the user. Nobody created a subject alternative name extension, nowadays often more important than the standard subject fields. Including it in the interactive mode, as GnuTLS does, may prove beneficial.

Certificate Version. In nearly three-quarters of cases (74%, 46/62 🎗) an X.509 version 3 certificate was created. The remaining participants (26%, 16/62 🎗) created an older certificate of version 1. A major difference is that the older version does not support any extensions (e.g., distinguishing between CA and user certificates, alternative names or key usage constraints). The trick to understanding this is the process of creation – if you generate a certificate in one go, version 3 certificate is created. On the other hand, if you split the process into generating a CSR and then signing it, you end up with an older (version 1) certificate.

Hash Function. All created certificates use SHA-256 as the underlying hash function. This result is slightly unexpected considering the number of lower-security 1024-bit keys and version 1 certificates.

Validity Period. The median validity of the created certificates is 1 year (60%, 37/62 🎗) and the range stretched from as short as 10 days to as long as 20 years (both 1%, 1/62 🎗). Similarly to keysize, none of the possible solutions require the user to specify the validity explicitly (default is 30 days, present in 23%, 14/62 🎗). Again, most of the people (79%, 49/62 🎗) did so (again, due to almost all examples and tutorials doing so). It is worth noting that 15% of the participants (9/62 🎗) created a certificate valid for 3 or more years in spite of knowing they are only creating a certificate for a momentary testing of a program feature.

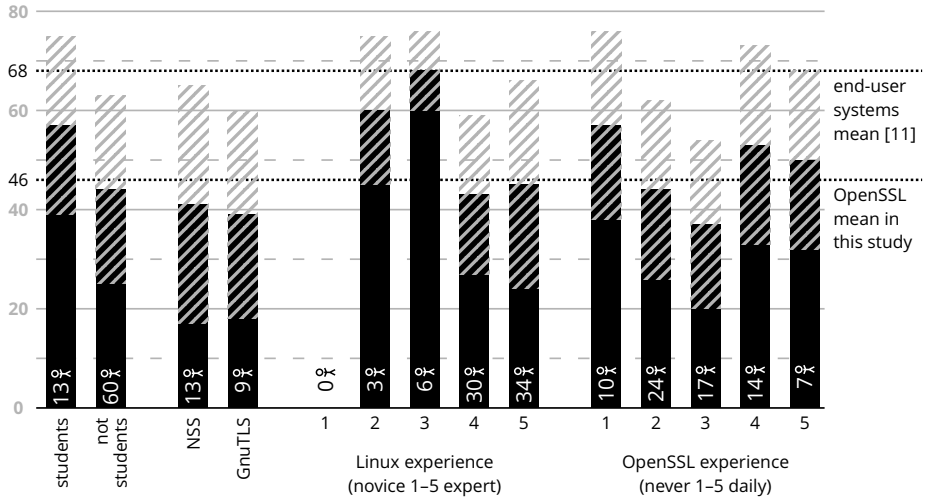

Fig. 3. SUS scores [11] for command line OpenSSL for different participant subgroups (73 users with relevant and complete answers, higher score means better usability).

4.3 Perceived Interface Usability

We assessed the perceived usability of command line OpenSSL in two ways: quantitatively using a standardized usability scale and qualitatively by the post-task interviews.

Usability Score. System usability scale (SUS) is a simple, standardized and widely applicable method to measure system usability. It records the level of agreement with 10 fixed statements regarding the user's experience. Even though the scale is not diagnostic (not exposing what exactly is wrong), it offers a straightforward comparison. The produced score lies between 0 and 100, with 68 being considered average for all end-user products or systems. [11]

The overall average score for command line OpenSSL was 46 (median 48), indicating a rather poor experience. We disregarded the opinion of users with incomplete SUS answers (5 ♀♂) and users who did not complete either of the tasks successfully (9 ♀♂). In the latter case, we could not guarantee their evaluation was related to OpenSSL (e.g., some created SSH keys).

The averages for different subgroups can be seen in Fig. 3. The score differs significantly between students (mean 57, 13/73 ♀♂) and not students (mean 44, 60/73 ♀♂). In addition, the usability score exhibits a small but significant negative correlation with the number of years the participant studied/worked in IT (Pearson correlation coefficient[7] $r = -0.26$). This suggests that the more years

[7] Pearson correlation coefficient r is interpreted similarly to Spearman's ρ, i.e., values from -1 to 1, the absolute value indicating intensity. [24]

Table 1. Summary of the most prominent participant opinions on OpenSSL and its manual page.

Command line tool	🧍	**Manual page**	🧍
It is too complex.	18	It is useless.	18
The interface is OK.	14	There are no examples.	18
The structure is confusing.	14	The structure is confusing.	16
Error messages are incomprehensible.	9	It is OK.	9
It is too low-level.	7	It is too long.	9
It does not follow Linux conventions.	7	It is only for experts.	7
It has inappropriate defaults.	4	There is no theory.	4

the participant has already spent studying/working in IT, the less usable he considers command line OpenSSL. The averages are smaller for participants with previous experience with other tools, though the difference is not significant – mean SUS score of 41 for NSS users (13/73 🧍) and 39 for GnuTLS users (9/73 🧍). All this suggests, rather counter-intuitively, that the usability of command line OpenSSL is lower for the users with more experience (not students, working in the field longer, knowing other tools).

As for the prior OpenSSL experience, the difference between the levels 1–5 (never–daily) was almost significant (ANOVA,[8] $p = 0.052$), although the relationship is not linear. The perceived usability is higher for those who have never used OpenSSL and those using it daily with the lowest score for users with average self-evaluated experience. This concurs with the previous observations (usability decreasing with experience), considering the fact that SUS score tends to increase as you use the system more [20].

The correlation with any other features measured in the pre-task questionnaire or the tasks success was not significant.

User Opinions. We have coded the post-task interviews to get basic insights into the participant opinions. The most frequent categories are summarized in Table 1. Only 16% of the users (14 🧍) expressed the opinion that the OpenSSL interface is generally OK (these participants also reported a significantly higher usability score when compared to the rest, but did not succeed significantly better). 21% (18 🧍) stated the tool was too complex and 16% (14 🧍) complained directly about the tool's structure being badly designed. Further objections included cryptic error messages, being too low-level, having inappropriate defaults or not following Linux conventions.

Interface Shortcomings. An interesting case, possibly related to the usability decrease with experience gain, comes with the complaints that OpenSSL does not follow Linux conventions. It does not provide any of the parameters `--help`, `-help`, `-h` nor a `help` subcommand (tried by 26 🧍, 10 🧍, 10 🧍 and 2 🧍, respectively).

[8] ANOVA is a method for comparing differences among groups of observations. [24]

The full-word command line options start with a single minus sign instead of the customary two (i.e., you need to use `-verbose` instead of `--verbose`). Furthermore, arguments order should not matter – currently, all options must precede the first non-option argument (e.g., one cannot add `-option` after specifying the first file to validate).

Sometimes, if a particular option is missing, OpenSSL assumes standard input. E.g., calling `openssl req` hangs the command without stating what is missing. Thus, instead of a useful error message, the user is left to figure out the error himself.

The existing error messages could be much more comprehensible. E.g., failing to set a passphrase produces a 3-line message saying the passphrase must be at least 4 characters long but also includes various memory addresses and function names. This particular error caused three participants to completely abandon the (correct) solution and search for a different one (even though the problem is stated clearly at the end of the first line). Another example is an argument typo producing a usage help but not stating at all what the problem is (e.g., `openssl verify -option`).

Two of our respondents got surprised by the set subject fields values, see Section 4.2). Furthermore, the default keysize should be unified – currently, creating a key through the `genrsa` or `req` modules results in 2048-bit key by default, while using the `genpkey` module (superseding `genrsa`) creates only a 1024-bit key.

These deviations from known good practices may seem small, but to objectively assess their effect on overall OpenSSL usability, a specialized experiment would be necessary.

4.4 Participant Behavior

We were surprised by several aspects of the participant behavior during task completion. Users looked into the created and validated certificates far less often than we expected. Only a quarter of the participants of the first task (25%, 22 🧍) did inspect their own result after creation and only a half of the users attempting the second task (50%, 36/72 🧍) displayed the contents of the provided certificates. In addition, participants sometimes totally ignored the produced error message (see Section 4.3).

In a few cases (9%, 8 🧍), the participants intentionally changed the parameters used in the tutorials/examples. In particular, they increased the keysize (8 🧍) and/or changed the proposed validity (2 🧍 increased, 2 🧍 decreased). These users did not differ from the rest of the respondents in any other aspect.

Some participants (28%, 24 🧍) answered the question on theoretical knowledge required for task completion. Of these, about a half (46%, 11/24 🧍) felt lacking such knowledge. However, the number is probably biased towards the negative answer, as people may have a tendency to emphasize what they lack to what they know.

During the work on experiment tasks, 7 participants (8%) took advantage of superuser privileges (using `sudo`). Two of them used it only when appropriate

(they interpreted the first task as generating a certificate for a new OS user Johnny they had to create). The rest (6%, 5 🧍) used the superuser privileges to browse OS-protected locations (system private SSL keys, system-wide trusted certificate store) or to run common OpenSSL commands. In two cases this was suggested by a tutorial – one generating SSL server certificates directly to protected webserver folders and the other just running all OpenSSL commands as the superuser for no apparent reason. While 5 participants are not many, using `sudo` unnecessarily is a clear security hazard.

At the end of the interview, 16% of the participants (14 🧍) proactively showed us other tools they use instead of interacting directly with OpenSSL. These were often higher-level tools or scripts building either directly on OpenSSL or on a similar backend with low-level capabilities. This supports the opinion that OpenSSL is too low-level to be used directly. Regarding the structure of the command line interface, the most frequent good example was the `ip` command (3/14 🧍). People liked its structure, context-aware help and context-aware autocomplete. These are all tool-agnostic features that could also be implemented for OpenSSL to support usable design.

4.5 Resources Used

We divided participants into four categories according to the resources they used while solving the task:

None	Neither going online nor browsing the manual pages.
Manual	Participants using local manual pages. Did not browse online.
Manual + online	Participants reading both local manual pages and online materials (tutorials, blogs, forums).
Online	Participants only browsing online, not using the manual pages.

Note that all people may have also used the command line help provided by OpenSSL. The distribution of used resources can be seen in Fig. 4. About half the participants used the combination of informal online sources (tutorials, blogs, forums) and official man pages installed locally (54%, 47 🧍 for Task

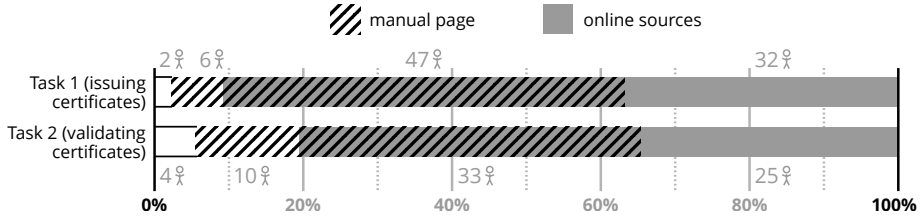

Fig. 4. Resources the participants used while completing the task (87 users issuing certificates, 72 users validating certificates).

1, 46%, 33/72 for Task 2). The second most prominent group was participants using only online resources. The resources used in the two tasks are correlated ($\rho = 0.33$), though not particularly strongly.

The correlation of used resources with task success was small and not significant. However, the presented order of resources significantly correlates with almost all answers on previous experience ($\rho \approx 0.3$), except for having a background in security. That is, the more years has the participant spent in IT (the more he knows about certificates or Linux, the more he has worked with certificates and OpenSSL), the less likely he is to use online resources (and the more likely to use local manual pages or nothing at all).

Online Sources. An overview of the websites visited during task completion is in Table 2, omitting four users with connection problems and pages not relevant to the task. From the 87 different second-level domains visited at least once, the most prominent was google.com (79 – all participants browsing online), with the search queries not surprising in any way (words `openssl`, `certificate`, `key`, `generate` and `public` dominating). The second most visited domain belonged to Stack Overflow[9](73%, 58/79). The remaining domains are a mixture of forums, public tutorial pages, private company pages and personal blogs. Sometimes the pages used were somewhat unexpected: the knowledge base of the University of Wisconsin-Madison was visited by 40% (29/72) of those attempting the second task (as it has a simple and straightforward page on certificate validation).

About half the participants (43) explicitly stated their trust in the Stack Overflow answers in the post-task interviews. About a half of them (53%, 23/43) indicated they trust answers found there. A third (33%, 14/43) mentioned that although they generally trust the answers, they always verify them elsewhere. The remaining 14% (6/43) declared they do not trust solutions from Stack Overflow or similar sites and do not use them.

To find out what user behavior was driven by the information in the online tutorials, we did a more detailed analysis of relevant websites visited by at least 2 participants (48 pages in total).

Nearly all solutions on these pages specified both the keysize and validity period using an explicit value, even though both have reasonable built-in defaults (2048 bits and 30 days). Only two Stack Overflow pages (out of 22) contained at least one solution using the default keysize and only one contained a solution setting the validity of "XXX days" (forcing the user to choose for himself, but avoiding the built-in default). This matches the actual user behavior (most of them unnecessarily stating the keysize and validity period explicitly, see Section 4.2). It hints that changing these OpenSSL defaults will probably have little effect. In the long term, teaching people that defaults are reliable can be both worthwhile (defaults evolve but tutorials usually do not) and a little dangerous (in case the defaults cease to be reasonable). The actual key lengths and validity

[9] Stack Overflow is a large online community for programmers to share their knowledge in a simple Q/A system, see stackoverflow.com.

Table 2. Relevant online pages visited by at least 2 grouped by domains (Sec. = Do pages contain security discussion?, Param. = Do pages explain individual parameters?).

Visitors	Domain	Site type	Pages	Sec.	Param.
100% 79	google.com	search engine	–	–	–
73% 58	stackoverflow.com	Q/A forum	7	◐	◐
39% 31	stackexchange.com	Q/A forum	4	◐	◐
38% 30	sslshopper.com	non-profit tutorial site	1	○	○
37% 29	wisc.edu	university tutorial site	1	○	○
20% 16	akadia.com	company support page	1	○	○
19% 15	openssl.org	official OpenSSL site	5	○	●
16% 13	digitalocean.com	company support page	1	○	●
13% 10	rietta.com	company support page	1	○	○
11% 9	cyberciti.biz	Q/A forum	1	○	○
11% 9	wikibooks.org	non-profit encyclopedia	1	○	○
10% 8	jamieLinux.com	personal blog	3	◐	○
10% 8	serverfault.com	Q/A forum	2	◐	◐
9% 7	asperasoft.com	company support page	1	○	○
9% 7	wikipedia.org	non-profit encyclopedia	1	●	○
8% 6	typo3.org	non-profit support page	1	○	○
6% 5	github.com	GIT repository provider	1	○	○
6% 5	msol.io	personal blog	1	○	○

periods used in the tutorials approximately match the results of the first task (the most prominent being 2048-bit keys and validity of about one year).

Most of the websites contained useful copy-pasteable code snippets (77%) and links to sites with further resources (73%). However, as can be seen in Table 2, only a few (23%) contained any security discussion (e.g., what are the risks of self-signed certificates, smaller keys or longer validity) and only 27% explained all the parameters used in the suggested code snippets. We see this as alarming, even though expected (e.g., see [8]). The security context may not have been relevant at the time of writing the particular tutorial/forum answer but may be crucial for the user visiting the site later (and possibly with a different use case). The absence of the parameter explanation often leads users to blindly try the proposed solution. This manifested itself also in the experiment, as people only rarely consulted the manual before executing the command.

One more fact concerns the official OpenSSL documentation online – 6 out of 13 relevant pages accessed at least once during the task completion did not exist. These pages were often linked in tutorials/forums since they represent the authoritative description of OpenSSL behavior. According to the data in the Internet Archive [1], the documentation changed structure without proper redirects at some point in 2016.

Manual Page. Only 17% of the participants (9) stated the manual page is OK. As for the negative opinions, 34% (18) complained the manual page contains

no examples (which is incorrect, the examples are further in the manuals), 30% (16 👤) said the manuals had a bad and/or confusing structure. Other objections included the manual being too long, being written for experts, lacking theory explanation or being generally useless, see Table 1 in Section 4.3.

The neglected examples in the manual pages could be solved by moving them to a more prominent position (i.e., higher up the page), although this would contradict the usual manual page structure.

One of the unexpected problems was to correctly invoke the appropriate manual page. Since OpenSSL is a complex tool, its manual is split into several independent pages named after the subcommands (i.e., to get a manual for `openssl x509` one has to, in Ubuntu, call `man x509`). 28% of the users (15 👤) wrongly called `man openssl <cmd>` with others trying also `man openssl-<cmd>` (4 👤) and `man openssl.<cmd>` (1 👤). To further complicate the matter, this behavior is OS-specific: Gentoo, for example, requires you to call for `man openssl-<cmd>` [3].

The main manual page (`man openssl`) should clearly note in the header that individual subcommands have separate manual pages available through `man <cmd>` (currently, there is no such notice). Regarding the syntax for manual invocation, adding simple symlinks for the intuitive variants would solve the problem easily (such symlinking has been already used, e.g., for git subcommands).

5 Study Limitations

The strongest limitation of this study is the self-selection bias of its participants – the research was open to all attendees of a large developer conference. The engaged user sample may thus not be representative of the wider developer community. Furthermore, the respondents may have behaved differently than if they were really at work. Firstly, the tasks were only hypothetical (there was no real software to pass the created certificate to). Secondly, they knew their efforts were recorded (the observer effect). Five participants (6%) even mentioned that they may have behaved differently was that a real situation.

Some of the technical aspects may be bound to the specific version of OpenSSL or the operating system. For example, the corresponding Fedora/RHEL OpenSSL package (1.0.2j-fips) has slightly different defaults, but they exhibit the same problems.

A limitation regarding the visited websites: We cannot say if the participants actually used them or deemed them useless after opening.

Lastly, part of the results depends on subjective evaluation the researchers. Website parameters (page relevance, the presence of the security discussion, parameter explanation), as well as interview coding, are subjective to the coder. Even though the tasks and questionnaire were precisely formulated (and also provided in writing), the answers to the interview questions may have slightly differed between the three researchers conducting interviews.

6 Related Work

Related usable security research falls into two categories: analysis of cryptographic interfaces (both user interfaces and application programmable interfaces – APIs) and documentation (both formal and informal).

Cryptographic Interfaces. Most of the usable security research examines the use cases of a "common Johnny" [14,23,27], not recognizing the situation of more knowledgeable users. A notable exception is a recent work by Krombholz et al. [17], focusing on the TLS configuration process. It concludes that the deployment process is far too complex even for people with proficient knowledge in the field.

Another analysis somewhat similar to ours was done by Georgiev et al. [15] for APIs, showing that SSL certificate validation is broken in many places (especially in non-browser software) due to unusable API design. Cryptographic APIs have also been heavily misused in Android with at least 88% applications having at least one API mistake [12]. In general, much more bugs seem to be misuses of the cryptographic software rather than problems of the libraries themselves [18].

There have been efforts to improve the interfaces (e.g., the Networking and Cryptography library [10]), but the empirical comparison of multiple libraries by Acar et al. [7] clearly shows that a usable interface does not suffice for a usable system. It also presents another relevant observation: X.509 certificate validation seems to be a more difficult task than both symmetric and asymmetric encryption. Apart from using the standardized system usability scale, the authors develop their own diagnostic usability scale that seems to be a viable alternative.

Research by Robillard [22] tries to identify reasons why interfaces are hard to learn, using a qualitative survey. The results are similar to outcomes of our interviews: deficiencies in structural design, uneasy debugging and documentation issues (mainly insufficient or inadequate examples). The work promotes the "principle of least astonishment", often seen violated by OpenSSL in our study (unexpected defaults, arguments not Linux-compliant, ...).

Documentation. Lethbridge et al. show [19] that software engineers do not update documentation much (except for testing and quality documentation). Nevertheless, they show that out-of-date documentation is still considered useful.

A survey by Uddin and Robillard [26] sheds light on documentation shortcomings: content causes more problems than presentation – the greatest difficulties are caused by incompleteness, ambiguity and bloat.

A paper by Fischer et al. [13] examines the impact of copy-pasting snippets from Stack Overflow on code security. They matched the extracted snippets to Android binaries, finding out that 15% contain copy-pasted code, 98% of which using at least one insecure snippet.

Acar et al. conducted an empirical study [8] investigating the impact of different information sources on code security. Developers allowed to use only Stack

Overflow produced less secure (although more functional) code than those with official documentation or books.

Based on such results, Subramanian et al. suggested to bridge the gap between formal and informal sources (official documentation and Stack Overflow) by adding interconnecting links to both places [25].

7 Conclusions

We conducted what we believe to be the first rigorous study of OpenSSL usability, aimed at attendees of a developer conference. In two tasks (generating and validating X.509 certificates), we observed participant success, use of resources, security-related behavior and collected their opinions.

The overall usability of OpenSSL turns out to be rather low (but probably still higher than other tools, as hinted by the pilot experiment). The low usability was also reflected in the high discrepancy between users' opinion of task success and reality. Moreover, we observed lower perceived usability for developers with more experience in the field.

About 20% of the created keys were only 1024-bit long, being a clear security concern. Furthermore, about a quarter of the created certificates were of version 1, lacking any extensions (alternative names, key usage constraints, etc.). On the other hand, all certificates used SHA-256, avoiding the deprecated SHA-1.

Both manual pages and online sources were used extensively, with Stack Overflow being accessed most often and also by most participants. It is worth noting that re-use of solutions/examples from online sources became a common developer practice, with consequences worth a further investigation.

Based on the observed behavior and user opinions, we suggest several improvements for the OpenSSL interface and its manual page. Small compatibility-preserving suggestions include consistent and secure defaults, better error messages, explicit note on the manual page split, symlinks for `man openssl <cmd>` and proper redirects for online manual pages. Bigger changes cover Linux compliant command line arguments, modification in the interactive certificate generation (e.g., an addition of subject alternative name extension) and more prominent display of examples in the manual page.

With help of OpenSSL developers, we already got alternative names for manual pages upstream, proposed a solution to the issue with missing web documentation redirects. Several other things have already improved in OpenSSL 1.1.0f independent of our research (e.g., there is now a `help` command and all commands support the `-help` argument). Incorporation of further changes requires a wider discussion in the developer community.

Further studies should be performed to establish the validity of our propositions (Does the command line argument format really matter? Do people really get discouraged by the current structure of the manual page?). Similar research should be done with other developer tools and other aspects of OpenSSL.

All in all, today's user-centered design must also acknowledge the usability issues present for knowledgeable users, not only those for the "common Johnny".

Acknowledgments. This work has been supported by Red Hat Czech and done in collaboration with Red Hat crypto team. We are particularly grateful to Nikos Mavrogiannopoulos and Jan Pazdziora for insightful ideas, to Lenka Horáková, Vlasta Šťavová and Agáta Dařbujánová for their help with the experiment and to Lujo Bauer and Martin Preisler for comments on the paper draft. Vashek Matyas thanks Red Hat Czech and CyLab, Carnegie Mellon University for a supportive sabbatical environment and the Czech Science Foundation project GBP202/12/G061 for partial funding. We also thank all experiment participants.

A Participant Questionnaire

Pre-Task Survey: Prior Knowledge and Experience

1. Do you know what public key certificates are and what they are used for? *(5-point scale from "never heard of it" to "work with them daily")*
2. How would you describe your experience with Linux OS? *(5-point scale from "novice Linux user" to "expert Linux user")*
3. How many years have you been studying+working in IT? *(number)*
4. What are your current positions? *(student/developer/quality engineering/IT analyst/tester/manager/quality assurance/documentation writer/other (please specify))*
5. Have you studied/worked specifically in IT security? *(5-point scale from "no security experience" to "security specialist")*
6. Have you ever generated or validated any public key certificates? *(5-point scale from "never" to "daily")*
7. Have you ever used 'openssl', the command line utility provided by OpenSSL? *(5-point scale from "never" to "daily")*
8. Have you ever used any other CLI tools for manipulating public key certificates? *(never/yes, but long ago/certutil (NSS)/certutil (Windows)/certtool (GnuTLS)/other (please specify))*

Task 1: Issuing Certificates

9. Have you been able to issue the certificate? *(yes/no/I don't know)*
   9a. If not or unsure: Please, describe briefly what went wrong.
10. How did it go? What did you do? Are you confident of what you've done?

Task 2: Validating Certificates

11. Have you been able to validate any certificates? *(yes/some/no/I don't know)*
12. Do you trust the certificates? *(for each certificate: yes/no (please specify reason)/I don't know)*
    12a. If not or unsure: Why are you unsure? Describe what happened.

Post-Task Interview: Your Experience with OpenSSL

13. Please fill in the attached System usability scale. *(5-point scale from "strongly agree" to "strongly disagree" for each of the 10 statements)*
14. What do you think of the interface of OpenSSL? Was it intuitive? Well-documented? Well-structured? Is there anything you would change?
15. Do you believe Stack Overflow solutions in general? Did you miss any theoretical knowledge during the task completion?

B Participant Quotations

Selected quotations from study participants are presented below to illustrate general feelings towards the library. However, the selection creates a somewhat biased impression – about 20% of the people considered both the tool interface and documentation fairly good considering the complexity of features it provides (though not expressing this strongly, with a single exception quoted below).

- "It's very humbling to have your tools taken away, be left with bare OpenSSL and not be able to fulfill simple tasks."
- "Interacting with OpenSSL *voluntarily*? Sorry, not even for research."
- "We all know it sucks, finally, there is someone collecting empirical data."
- "OpenSSL? I hate every single bit of it."
- "Working with OpenSSL is a struggle every time – it takes at least 20-30 minutes to find something."
- "The person writing the manual page has much different use cases than the person reading the manual."
- "I am surprised that even as a crypto expert I am unable to use OpenSSL."
- "The manual page presumes you know what you are doing."
- "You need to know crypto, ASN.1, X.509 and C to be able to use OpenSSL correctly."
- "OpenSSL is like a set of sharp knives."
- "OpenSSL is disgustingly complicated. I always spend half a day reading and googling."
- "OpenSSL is intuitive and well documented, I wouldn't change anything."

References

1. Internet Archive: Wayback Machine, archive.org/web
2. Java Keytool, docs.oracle.com/javase/9/tools/keytool.htm
3. Man page search on Gentoo, www.polarhome.com/service/man/?of=Gentoo
4. Network Security Services, developer.mozilla.org/docs/Mozilla/Projects/NSS
5. OpenSSL: Cryptography and SSL/TLS Toolkit, www.openssl.org
6. The GnuTLS Transport Layer Security Library, www.gnutls.org
7. Acar, Y., Backes, M., Fahl, S., Garfinkel, S., Kim, D., Mazurek, M., Stransky, C.: Comparing the Usability of Cryptographic APIs. In: 2017 IEEE Symposium on Security and Privacy. IEEE (2017)
8. Acar, Y., Backes, M., Fahl, S., Kim, D., Mazurek, M., Stransky, C.: You Get Where You're Looking For: The Impact Of Information Sources on Code Security. In: 2016 IEEE Symposium on Security and Privacy. pp. 289–305. IEEE (2016)
9. Barker, E., Dang, Q.: NIST SP 800-57 Recommendation for Key Management Part 3: Application-Specific Key Management Guidance. Tech. rep. (2015)
10. Bernstein, D., Lange, T., Schwabe, P.: The Security Impact of a New Cryptographic Library. In: Proceedings of the 2nd International Conference on Cryptology and Information Security in Latin America. pp. 159–176. Springer (2012)
11. Brooke, J.: SUS – A Quick and Dirty Usability Scale. Usability Evaluation in Industry 189(194), 4–7 (1996)

12. Egele, M., Brumley, D., Fratantonio, Y., Kruegel, C.: An Empirical Study of Cryptographic Misuse in Android Applications. In: Proceedings of the 2013 ACM Conference on Computer and Communications Security. pp. 73–84. ACM Press (2013)
13. Fischer, F., Bottinger, K., Xiao, H., Stransky, C., Acar, Y., Backes, M., Fahl, S.: Stack Overflow Considered Harmful? The Impact of Copy&Paste on Android Application Security. In: 2017 IEEE Symposium on Security and Privacy. IEEE (2017)
14. Garfinkel, S., Miller, R.: Johnny 2: A User Test of Key Continuity Management with S/MIME and Outlook Express. In: Proceedings of the 2005 Symposium on Usable Privacy and Security. pp. 13–24. ACM Press (2005)
15. Georgiev, M., Iyengar, S., Jana, S., Anubhai, R., Boneh, D., Shmatikov, V.: The Most Dangerous Code in the World: Validating SSL Certificates in Non-Browser Software. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security. pp. 38–49. ACM Press (2012)
16. Horáková, L.: User Interface Design for Certificate Operations with Network Security Services. Master thesis, Masaryk University (2017)
17. Krombholz, K., Mayer, W., Schmiedecker, M., Weippl, E.: "I Have No Idea What I'm Doing" – On the Usability of Deploying HTTPS. In: Proceedings of the 26th USENIX Security Symposium. USENIX Association (2017)
18. Lazar, D., Chen, H., Wang, X., Zeldovich, N.: Why does cryptographic software fail? In: Proceedings of 5th Asia-Pacific Workshop on Systems. pp. 7:1–7:7. ACM Press (2014)
19. Lethbridge, T., Singer, J., Forward, A.: How Software Engineers Use Documentation: The State of the Practice. IEEE Software 20(6), 35–39 (2003)
20. McLellan, S., Muddimer, A., Peres, C.: The Effect of Experience on System Usability Scale Ratings. Journal of Usability Studies 7(2), 56–67 (2012)
21. Nemec, M., Klinec, D., Svenda, P., Sekan, P., Matyas, V.: Measuring Popularity of Cryptographic Libraries in Internet-Wide Scans. In: Proceedings of the 33rd Annual Computer Security Applications Conference. pp. 162–175. ACSAC 2017, ACM (2017)
22. Robillard, M.: What Makes APIs Hard to Learn? Answers from Developers. IEEE Software 26(6), 27–34 (2009)
23. Sheng, S., Broderick, L., Koranda, C., Hyland, J.: Why Johnny Still Can't Encrypt: Evaluating the Usability of Email Encryption Software. In: Proceedings of the 2006 Symposium On Usable Privacy and Security. pp. 3–4. ACM Press (2006)
24. Sheskin, D.: Handbook of Parametric and Nonparametric Statistical Procedures. Chapman and Hall/CRC, 4 edn. (2007)
25. Subramanian, S., Inozemtseva, L., Holmes, R.: Live API documentation. In: Proceedings of the 36th International Conference on Software Engineering. pp. 643–652. ACM Press (2014)
26. Uddin, G., Robillard, M.P.: How API Documentation Fails. IEEE Software 32(4), 68–75 (2015)
27. Whitten, A., Tygar, J.: Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0. In: Proceedings of the 8th USENIX Security Symposium. vol. 8, pp. 169–184. USENIX Association (1999)